

3. Ativar o BMP no FRR

1. Adicionar configuração do BMP no FRR

```
bmp targets gobmp
  bmp stats interval 5000
  no bmp stats send-experimental
  bmp monitor ipv6 unicast pre-policy
  bmp monitor ipv6 unicast post-policy
  bmp monitor ipv6 unicast loc-rib
  bmp connect 3fff:f3::1 port 5000 min-retry 120 max-retry 240 source-interface
ens5
```

4. Copiar diretórios bmp-stack e mudar permissão

```
cp -r /home/ceptro/Downloads/bmp-stack /opt/

sudo chown -R "$USER":"$USER" /opt/bmp-stack
```

5. Configuraração Kafka

1. Criar Kafka Cluster ID

```
KAFKA_CLUSTER_ID=$(
  sudo docker run --rm \
    --entrypoint /opt/kafka/bin/kafka-storage.sh \
    apache/kafka:4.3.1 random-uuid
)

printf 'KAFKA_CLUSTER_ID=%s\n' "$KAFKA_CLUSTER_ID"
```

2. Inserir o Kafka Clust ID no .env

3. Iniciar e validar o Kafka

```
sudo docker compose config -q

sudo docker compose up -d kafka

sudo docker compose ps

sudo docker compose logs --tail=50 kafka
```

4. Iniciar o container kafka-init e criar os tópicos

```
sudo docker compose run --rm kafka-init
```

5. Validar a criação dos tópicos

```
sudo docker compose exec kafka \
  /opt/kafka/bin/kafka-topics.sh \
  --bootstrap-server kafka:9092 \
```

```
--list
```

6. Verificar se o kafka responde na porta 19092

```
sudo docker run --rm \
  --network host \
  apache/kafka:4.3.1 \
  /opt/kafka/bin/kafka-topics.sh \
  --bootstrap-server '[::1]:19092' \
  --list
```

6. Configurar a conexão goBMP → Kafka

1. Criar config.yaml

```
bmp_listen_port: 5000

split_af: true

kafka_config:
  kafka_srv: "[::1]:19092"
  kafka_tp_retn_time_ms: 604800000
  skip_topic_creation: true
```

2. Iniciar o goBMP utilizando o arquivo de configuração

```
./bin/gobmp --config=/etc/gobmp/config.yaml --v=3
```

3. Verificar as mensagens no Kafka

```
sudo docker compose exec kafka \
  /opt/kafka/bin/kafka-console-consumer.sh \
  --bootstrap-server kafka:9092 \
  --topic gobmp.parsed.peer \
  --from-beginning \
  --max-messages 5 \
  --timeout-ms 10000

sudo docker compose exec kafka \
  /opt/kafka/bin/kafka-console-consumer.sh \
  --bootstrap-server kafka:9092 \
  --topic gobmp.parsed.unicast_prefix_v6 \
  --from-beginning \
  --max-messages 10 \
  --timeout-ms 10000

sudo docker compose exec kafka \
  /opt/kafka/bin/kafka-console-consumer.sh \
```

```
--bootstrap-server kafka:9092 \  
--topic gobmp.parsed.statistics \  
--from-beginning \  
--max-messages 10 \  
--timeout-ms 10000
```

7. Configurando o Postgres

1. Iniciar o container do Postgres

```
sudo docker compose up -d postgres
```

2. Verificar o container e a tabela bmp_events

```
sudo docker compose ps  
  
sudo docker compose exec postgres \  
psql -U bmp -d bmp -c '\dt'  
  
sudo docker compose exec postgres \  
psql -U bmp -d bmp -c '\d bmp_events'
```

8. Configurando o Redpanda Connect

1. Iniciar o container do Redpanda Connect

```
sudo docker compose pull connect  
  
sudo docker compose run --rm connect \  
lint /connect.yaml  
  
sudo docker compose up -d connect  
  
sudo docker compose logs --tail=50 connect
```

9. Verificar mensagens recebidas no Postgres

1. Contar mensagens recebidas

```
sudo docker compose exec postgres \  
psql -U bmp -d bmp -P pager=off -c \  
"SELECT  
    topic,  
    count(*) AS events,  
    min(kafka_offset) AS first_offset,  
    max(kafka_offset) AS last_offset  
FROM bmp_events  
GROUP BY topic  
ORDER BY topic;"
```

2. Exibir o JSON original

```
sudo docker compose exec postgres \
psql -U bmp -d bmp -P pager=off -x -c \
"SELECT
    id,
    topic,
    kafka_offset,
    payload_raw
FROM bmp_events
WHERE topic='gobmp.parsed.unicast_prefix_v6'
ORDER BY id DESC
LIMIT 1;"
```

3. Exibir versão 'readable'

```
sudo docker compose exec postgres \
psql -U bmp -d bmp -P pager=off -Atc \
"SELECT jsonb_pretty(payload)
FROM bmp_events
WHERE topic='gobmp.parsed.unicast_prefix_v6'
AND payload->>'action'='add'
ORDER BY id DESC
LIMIT 1;"
```

10. Normalização dos dados

1. Parar o Redpanda

```
sudo docker compose stop connect
```

2. Verificar consumer group no Kafka

```
sudo docker compose exec kafka \
/opt/kafka/bin/kafka-consumer-groups.sh \
--bootstrap-server kafka:9092 \
--describe \
--group bmp-postgres-raw-v1
```

3. Aplicar os novos *schemas*

```
(
set -e

sudo docker compose exec -T postgres \
psql -v ON_ERROR_STOP=1 -U bmp -d bmp \
< postgres/002-routing-normalization.sql

sudo docker compose exec -T postgres \
```

```
psql -v ON_ERROR_STOP=1 -U bmp -d bmp \  
< postgres/003-statistics-normalization.sql
```

```
sudo docker compose exec -T postgres \  
psql -v ON_ERROR_STOP=1 -U bmp -d bmp \  
< postgres/004-backfill.sql
```

```
)
```

4. Verificar as tabelas e views criadas

```
sudo docker compose exec postgres \  
psql -U bmp -d bmp -c '\dt'
```

```
sudo docker compose exec postgres \  
psql -U bmp -d bmp -c '\dv'
```

5. Exemplo de consultas

5.1 Peers Ativos

```
sudo docker compose exec postgres \  
psql -U bmp -d bmp -P pager=off -c \  
"SELECT  
    speaker_id,  
    host(peer_ip) AS peer,  
    peer_asn,  
    state  
FROM peers_current  
ORDER BY peer_ip;"
```

5.2 Rotas nas RIBs (post/pre/loc)

```
sudo docker compose exec postgres \  
psql -U bmp -d bmp -P pager=off -c \  
"SELECT  
    rib_type,  
    COALESCE(host(peer_ip), 'LOC-RIB') AS peer,  
    count(*) AS routes  
FROM routes_current  
WHERE speaker_id='rt-borda'  
GROUP BY rib_type, peer_ip  
ORDER BY rib_type, peer;"
```

5.3 Verificar as estatísticas

```
sudo docker compose exec postgres \  
psql -U bmp -d bmp -P pager=off -c \  
"SELECT
```

```
    host(peer_ip) AS peer,  
    metric_name,  
    metric_value,  
    metric_kind,  
    kafka_time  
FROM bmp_statistics_latest  
WHERE speaker_id='rt-borda'  
ORDER BY peer_ip,metric_name;"
```

11. Mudar configuração no Redpanda Connect

1. Alterar a *query* no connect.yaml

```
query: |  
    SELECT ingest_bmp_pipeline_event(  
        $1, $2, $3, $4, $5, $6  
    )
```

2. Reiniciar o Redpanda Connect

```
sudo docker compose run --rm connect \  
    lint /connect.yaml  
  
sudo docker compose up -d connect  
  
sudo docker compose logs --tail=50 connect
```

3. Verificar os eventos normalizados

```
sudo docker compose exec postgres \  
    psql -U bmp -d bmp -P pager=off -c \  
    "SELECT  
        id,  
        topic,  
        kafka_offset,  
        normalization_status,  
        normalization_error  
    FROM bmp_events  
    ORDER BY id DESC  
    LIMIT 12;"
```

12. Configurando o Grafana

1. Carregar as variáveis de ambiente

```
set -a  
source .env  
set +a
```

2. Executar *schema* do usuário grafana

```
sudo docker compose exec -T postgres \
  psql \
  -v ON_ERROR_STOP=1 \
  -v grafana_password="$GRAFANA_DB_PASSWORD" \
  -U bmp \
  -d bmp \
  < postgres/005-grafana-role.sql
```

3. Iniciar o grafana

```
sudo docker compose pull grafana

sudo docker compose up -d grafana

sudo docker compose ps

sudo docker compose logs --tail=50 grafana
```

4. Configurar o Postgres no grafana

```
URL: postgres:5432
Database Name: bmp
Username: Grafana
Password:
TLS/SSL Mode: Disable
```

5. Teste do datasource

```
SELECT
  rib_type,
  COALESCE(host(peer_ip), 'LOC-RIB') AS peer,
  count(*) AS routes
FROM routes_current
WHERE speaker_id='rt-borda'
GROUP BY rib_type, peer_ip
ORDER BY rib_type, peer;
```